Device Tree

The Disaster so Far

Mark Rutland
<mark.rutland@arm.com>

ELC Europe 2013

Agenda

Disaster is a strong word. Let's talk about:

- What was wrong with board files
- What device tree is (and what it isn't)
- The ARM conversion so far
- The problems we have, and how to fix them
- What we need to do in future

Where we came from

Two big problems:

- Hard-coded board description
  - Kernel must know every possible configuration
  - Minor revisions require a new kernel
- Separate kernels per platform
  - Uncoordinated – "stepping on each others toes"
  - Difficult to test
  - Painful for distributions

Planned solution:

- Single image
- Dynamic configuration
- Move board description out of the kernel

Device Tree – Overview

- A data structure for describing hardware
- Defined by OpenFirmware (IEEE1275-1994)
- Extended by ePAPR & other supplements
- Handled by OpenFirmware, U-Boot, ...
- Used by *BSD, Linux, OS X, Solaris, Xen
- Used on ARC, ARM(64) C6X, META, MicroBlaze, MIPS, OpenRISC, PowerPC, SPARC, x86, Xtensa
- Generated by KVM tool, Xen, others

Device Tree – Overview

- Declarative hardware description
  - Describes hardware topology
  - Format not tied to OS internals
  - Hierarchical key value(-list)
- Just a data structure
  - Conventions, not rigid rules
- Bindings
  - Conventions for describing a particular devices
  - Typically matched by compatible string
  - Device classes share common bindings
- No central authority
  - Bindings created by users
  - No coordination of implementors

Device Tree – Bindings

```
Vendor dev2000 bindings
=======================
The Vendor dev2000 is a memory-mapped device that
may or may not do something useful. V2 dev2000s
support the v1 programming interface.

Required properties
-------------------
- compatible: should contain:
    * "vendor,dev2000-v2" for v2 devices.
    * "vendor,dev2000" for v1 or v2 devices.
- reg: offset and length of the registers.
- interrupts: should contain interrupt-specifiers
              for DEVINTR and DEVINTR2.
```

Device Tree – Source

```
#address-cells = <1>;
#size-cells = <1>;
ic: ic {
        compatible = "vendor,standard-ic";
        interrupt-controller;
        #interrupt-cells = <2>;
};
dev: device@0xffff7000 {
        reg = <0xffff7000 0x4000>;
        compatible = "vendor,dev2000-v2",
                     "vendor,dev2000";
        interrupt-parent = <&ic>;
        interrupts = <17 33>, <11 47>;
};
```

Unfamiliarity

- Device tree is novel to many of us
  - History & idioms not well known
  - Undocumented assumptions
- Documentation difficult to find
  - OpenFirmware.org no longer online
  - playground.sun.com no longer online
  - IEEE 1275 difficult to find
- Remaining documentation not always helpful
  - Binding documents often inconsistent / vague
  - No clear right way to do things

Inconsistency

How do we refer to interrupts?

- Interrupt connection
- The single IRQ line
- Interrupt source of the parent interrupt controller
- One interrupt to each core
- Interrupt mapping for XXXX IRQ
- Interrupt number to the cpu
- Standard interrupt property
- An interrupt node describing the IRQ line
- ...

Get acquainted with device tree

- ePAPR still online
- Linux documentation & source still available
- Ongoing effort to standardise bindings
  - Look for bindings reviewed by device tree maintainers
- Planned effort to improve documentation
  - Binding review checklist
  - Designing future-proof bindings
  - Schemas
  - eAAPR?
- `devicetree@vger.kernel.org`
- Freenode `#devicetree`

We are used to board files

- Compiled into kernel
  - Atomic updates
  - Describe what Linux wants to know now
    - Subset of hardware
    - Policy
  - What documentation...?
- Conversion to dt looks simple
  - `platform_device::name` $\mapsto$ compatible
  - `IORESOURCE_MEM` $\mapsto$ reg
  - `IORESOURCE_IRQ` $\mapsto$ interrupts

Cleanup is breakage

```
From 365594088a123609a6cd454fa5a60b46b1423cd3 Mon Sep 17 00:00:00 2001
From: Joe Developer <joe.developer@vendor.com>
Date: Tue, 15 Oct 2013 23:25:56 +0100
Subject: [PATCH] ARM: platform: change some existing compatible string

We have a new hardware revision, and "vendor,device" isn't general
enough. Replace "vendor,device" with "vendor,device-xxxSOCVARIANTyyy",
and introduce an entirely new naming scheme.

Signed-off-by: Joe Developer <joe.developer@vendor.com>
---
 arch/arm/boot/dts/vendor-platform.dtsi |   28 ++++++++++++++++++-------------
 drivers/sys/vendor-device.c            |    2 +-
 ../devicetree/binding/arm/somedev.txt  |    2 +-
 3 files changed, XX insertions(+), YY deletions(-)
```

Device tree is an ABI

- Device tree is in use **now**
  - Products shipping with it
  - Users expect it to work
  - Other developers want it to work
- Once working, a DT should not require changes
  1. Device trees describe hardware
  2. The hardware doesn't change
  3. Required changes are a regression
- We are not omniscient
  - Bindings can be extended
  - New bindings can be introduced
  - Old bindings must still work
  - Staged deprecation

We make mistakes

```
- clocks : From common clock binding. First clock is
    phandle to clock for apb pclk. Additional clocks
    are optional and specific to those peripherals.
- clock-names : From common clock binding. Shall be
    "apb_pclk" for first clock.
```

```
mmci@050000 {
        compatible = "arm,pl180", "arm,primecell";
        clocks = <&v2m_clk24mhz>, <&smbclk>;
        clock-names = "mclk", "apb_pclk";
}
```

Design for extension and correction

- Be precise
  - Avoid ambiguity
  - Define specific compatible strings
  - Support named resources
  - Describe property types
- Enable description of all resources
  - Read the manual, not BSP
  - One clock $\implies$ all clocks
  - Describe the whole register bank
- Consider the future
  - Will the next version have REFCLK?
  - What if `#interrupt-cells` grows?
  - Parsing notes

The conversion process

Top Down

1. Start with board files
2. Tear down until empty
3. Deprecate board files

- Board files fill gaps
- Works immediately
- DT changes required
- Problems apparent late

Bottom Up

1. Start with blank slate
2. Build up to full platform
3. Deprecate board files

- Must describe everything
- Long lead time
- Once working, likely stable
- Problems apparent earlier

A fresh start: mach-virt

- Empty (virtual) machine descriptor – no platform code
- All devices instantiated from device tree
- SMP without platform code (with PSCI)
- Used by KVM & Xen
- Where possible, start here

```
/ {
        compatible = "vendor,platform",
                     "linux,dummy-virt";
        /*
         * Anything you want here...
         */
};
```

Binding review

- Drinking from the firehose
  - Few reviewers
  - Lots of binding authors
  - Lots of trivial issues
  - A bottleneck
  - Documentation mingled with code
  - Novel devices and subsystems
- We are not universal experts
  - Missing/incorrect details missed
  - Need help from maintainers
- Getting better
  - DT becoming more familiar
  - Bindings classes have established patterns

Better binding review

- Established subsystems well-understood
  - Don't be needlessly different
  - Maintainers trusted to review bindings
- Help us to help
  - What is this device?
  - Link to documentation
  - Why do you need this property?
  - Join in the review
- Be explicit
  - Define property types
  - Refer to other bindings

Missed opportunity – We're not sharing

FreeBSD:

```
compatible = "arm,gic";
```

Linux:

```
compatible = "arm,cortex-a9-gic";
```

- We could have common bindings
- We must cooperate with other device tree users
  - Ensure generality of bindings
  - Ensure compatibility
  - Share burden – free DTs
- Cannot pretend we're in charge

ACPI is on the horizon

But:

- Very few ARM community members with ACPI experience
- Almost all DT problems applicable
- Do we want to repeat the same set of mistakes?

Let's do it right from the start:

- No crutches – everything in ACPI
- Describe the hardware, not today's usage
- Design for the future
- Cooperate with other OS communities

How to help

- **Describe the hardware** not its use
  - Gives the OS more flexibility
  - Encourages extensible description
- **Plan ahead** – *you* know what about future hardware
  - Consider how bindings must be extended
  - Raise problems with frameworks *now*
- **Work with others**
  - More eyes means fewer bugs
  - Easier to support long-term
  - Help others to help!
- **Be proactive** – report (and fix) problems
  - Fix issues today – lesser burden later
  - If a binding is broken, don't work around it

Thanks for listening

Questions?

Thanks for listening